CS5205 Complexity

Paul Ahern

May 31, 2007

Abstract

Notes from lectures.

Contents

1 Key Facts to Know **2**
1.1 Tree decomposition 2

2 Lecture 17 January 2007 **3**
2.1 Complex Systems 3

3 Lecture 18 January 2007 **3**
3.1 Process of modelling a Complex System 4
3.2 The system design process 5

4 Lecture 24 January 2007 **5**
4.1 System Design 5
4.2 Small World Graphs 6

5 Lecture 25 January 2007 **6**
5.1 Models for Complex Systems 6
5.2 Modelling of Dynamic Systems 7
5.3 Functional Classes 7

6 Lecture 31 January 2007 **8**
6.1 Example Model 9

7 Lecture 1 February 2007 **9**
7.1 History 9
7.2 Diagnostic Problem Solving 10

8 Lecture 7 February 2007 **10**

9 Lecture 8 February 2007 **10**

10 Lecture 14 February 2007 **11**

11 Lecture 15 February 2007 **11**
11.1 Model-Based Diagnosis 11
11.1.1 Probability-Based 12

12 Lecture 28 February 2007 **12**
12.1 Bond Graph 12
12.2 Modelling 13
12.2.1 Bond Graph Elements 13

13 Lecture 14 March 2007 **14**
13.1 Graphs 14
13.2 Complexity 16

14 Lecture 15 March 2007 **16**

15 Lecture 21 March 2007 **17**
15.1 Bayesian Inference 17

16 Lecture 26 March 2007 **18**
16.1 Tree decomposition 18

1 Key Facts to Know

A complex system is a system whose properties are not fully explained by an understanding of its component parts.

Complexity The essence of which is the property of self-organisation or emergence of structure from the interaction between the constituent parts of the system.

Model-Based Diagnosis Use a model of the system to reason about the causes of observed faults.

Bond Graph Energy-based graphical description of a dynamic system.

Complexity P problems that can be solved in polynomial time. **NP** problems for which a solution can be verified in polynomial time. It is unknown whether $P = NP$ (most suspect not).

1.1 Tree decomposition

Convert an arbitrary graph into a tree (of cliques).

1. Moralisation, marry (link) unconnected parents.
2. Triangulation, in any cycle of length greater than three, every triple must have a chord. Create a set of triangles.
3. Identify cliques.
4. Generate Clique tree, maintain the running intersection property.

2 Lecture 17 January 2007

Website www.cs.ucc.ie/~gprovan/CS5205/CS5205-ComplexSystems.htm

Grading 15% for mid-term presentation; 50% for project; 35% for summer exam.

This course will provide an introduction to modelling of complex systems; It will emphasise the practical over the theoretical.

Some coherent mathematical principles underlie the structure of many complex systems: e.g. Small World Graph.

Their behaviour differs according to many modelling frameworks (e.g. blood-flow in the body v packet flow around the Internet).

The course will introduce many software tools to aid in modelling.

2.1 Complex Systems

Complex Systems arise at the boundary between deterministic and chaotic phenomena.

Complex Systems can be described at multiple levels (at many scales). Made of many non-identical elements connected by diverse interactions (underlying graphical structure).

A complex system is a system whose properties are not fully explained by an understanding of its component parts.

Emergence "the arising of novel and coherent structures, patterns and properties during the process of self-organization in complex systems". (Goldstein 1999)

The overall emergent behavior is difficult to predict, even when subsystem behavior is readily predictable. Small changes in inputs or parameters may produce large changes in behavior.

Butterfly Effect Sensitive dependence on initial conditions.

Complex network forms the backbone of complex systems: the nodes correspond to the agents, entities or parts of the complex system, the edges to the interactions between them. The most complex networks are small-world or scale-free networks at the border between regular and random networks.

3 Lecture 18 January 2007

Complex Systems have historically been analysed on the basis of vertical decomposition (e.g. the Internet in terms of the protocol layer model - HTTP atop TCP atop IP). This course will also look at systems in terms of horizontal decomposition (e.g. analysing at the topography of the router network graph of the Internet) and relate the two types of analysis to one another.

Models are hierarchically decomposable. For example, looking at the electricity generation and distribution system, one might start with an abstract model (e.g. economic model concerned merely with supply and demand). Then again the graph of the distribution network might be relevant to the question at hand (e.g. optimising throughput or ensure that the system is robust).

3.1 Process of modelling a Complex System

There is a generic methodology in the study of complex systems. We will study Discrete Event and Continuous models of systems (and the Hybrid models which seek to bridge the gap between them).

- Understanding how a complex systems function depends upon detective work.
  - System participants often themselves don't understand how the system operates
  - As an outsider you won't either
- Accordingly, you will almost never start the complex system analysis process with the information needed to develop a mathematical representation.

The process of analysis will be:

1. Determine what questions you are seeking to answer with this model.
2. Develop from abstract to detailed models of the system.
   The objective is to decompose the problem into tractable entities which can be decomposed into Precise Mathematical Specifications.

- Representation is a key component in making sense of a complex system.
- Inevitable that cognitive models of a system are created, hence formal representation necessary to ensure that these cognitive models are: Verifiable, Shared and Accurate.

No single representation can adequately describe a complex system.

- Model to study operation of complete system
  As opposed to operation of the individual parts
- Method of model building: compositionality
- Method of analysis: isolate into parts
- Unified approach to modeling, rather than being domain-specific
  Qualitative reasoning schemes based on (incomplete) analytic models of system
  Energy-based modeling of physical systems
  Discrete-event models of systems (change in system directly linked to the occurrence of events)
  Combined modeling paradigms (Hybrid Systems approach to modeling)

What is needed is set of visual models, each describing different facets of the same complex system so that all relevant information can be captured and used. We can't discard information because we don't know what matters yet to define a mathematical model.

3.2 The system design process

- Partition requirements
  - Organise requirements into related groups.
- Identify sub-systems
  - Identify a set of sub-systems which collectively can meet the system requirements.
- Assign requirements to sub-systems
  - Causes particular problems when COTS (Commercial Off-The-shelf Software?) are integrated.
- Specify sub-system functionality.
- Define sub-system interfaces
  - Critical activity for parallel sub-system development.

4 Lecture 24 January 2007

4.1 System Design

Process:

1. Partition Requirements
2. Identify sub-systems
3. Assign requirements to sub-systems
4. Specify sub-system functionality
5. Define sub-system interfaces

Repeatedly iterate up and down this process. Use the Spiral model of requirements/design

Modelling:

- An architectural model presents an abstract view of the sub-systems making up a system
- May include major information flows between sub-systems
- Usually presented as a block diagram
- May identify different types of functional component in the model

System Integration

- The process of putting hardware, software and people together to make a system.

- Should be tackled incrementally so that subsystems are integrated one at a time.
- Interface problems between sub-systems are usually found at this stage.
- May be problems with uncoordinated deliveries of system components.

4.2 Small World Graphs

A small-world graph (or network) is a class of random graphs where most nodes are not neighbors of one another, but most nodes can be reached from every other by a small number of hops or steps. A small world network, where nodes represent people and edges connect people that know each other, captures the small world phenomenon of strangers being linked by a mutual acquaintance.

Many empirical graphs are well modeled by small-world networks. Social networks, the connectivity of the Internet, and gene networks all exhibit small-world network characteristics.

A certain category of small-world networks were identified as a class of random graphs by Duncan Watts and Steven Strogatz in 1998. They noted that graphs could be classified according to their clustering coefficient and their mean-shortest path length. While many random graphs exhibit a small shortest path (varying typically as the logarithm of the number of nodes) they also usually have a small clustering coefficient. Watts and Strogatz measured that in fact many real-world networks have a small shortest path but also a clustering coefficient significantly higher than expected by random chance. Watts and Strogatz proposed a simple model of random graphs with (i) a small average shortest path and (ii) a large clustering coefficient. The first description of the crossover in the Watts-Strogatz model between a "large world" (such as a lattice) and a small-world was described by Barthelemy and Amaral in 1999. This work was followed by a large number of studies including exact results (Barrat and Weigt, 1999; Dorogovtsev and Mendes)

Several procedures are known to generate small-world networks from scratch. One of these methods is known as preferential attachment. In this model, new nodes are added to a pre-existing network, and connected to each of the original nodes with a probability proportional to the number of connections each of the original nodes already had. I.e., new nodes are more likely to attach to hubs than peripheral nodes. Statistically, this method will generate a power-law distributed small-world network.

5 Lecture 25 January 2007

5.1 Models for Complex Systems

Topological Models share the following attributes:

- Clustering (densely connected regions loosely connected to each other).
- Short path-lengths.
- Scale-free (same properties maintain at all scales)
- Geographic/Spacial distribution (constrained in 2D or 3D).

Models:

1. Random Graph.
2. Small World Graph (Watts/Strogatz).
3. Preferential Attachment (Barabasi/Albert).
4. Kleinberg - Network Routing (aims for log n computation of routing).
5. Newest: Bipartite, Geographical, etc.

The first three types are most common in the literature.

Much more work needs to be done to model complex systems, in terms of their topology. Tens of papers are published every day. Most are from the Mathematics and Physics community and are based on tractable and elegant models.

G. Provan believes that an approach based on optimisation (in terms of functionality) is more likely to succeed.

These models have a small set of parameters. However every application domain has slightly different properties. Functionality drives different topologies, e.g. types of circuit layouts - Discrete Combinatorial Circuits (without feedback), Combinatorial Circuits with feedback, Complex Integrated Circuits, Analog circuits, etc.

5.2 Modelling of Dynamic Systems

State-Determined Systems: Our goal is to start with physical component descriptions of systems understanding of component behavior to create mathematical models of the system.

Mathematical model of state-determined system: Defined by set of ordinary differential equations on the so-called *state variables*. Algebraic relations define values of other system variables to state variables.

Dynamic behavior of state-determined system: Defined by (i) values of state variables at some initial time, and (ii) future time history of input quantities to system.

In other words, our system models satisfy the Markov property.

5.3 Functional Classes

Model Based systems and diagnosis. Model to study operation of complete system as opposed to the operation of individual parts.

Discrete and Continuous approaches traditional, now more Hybrid systems view being developed.

Energy-Based modelling of systems:

In the 1970s Qualitative Abstractions were developed. Leading to Qualitative Physics and Constraint Satisfaction studies in the 1980s. For example the water-level model.

I is the inflow of water into the tank. O is the outflow. L is the water level (L is continuous).

$\frac{dL}{dt}$ (water level over time) has domain: {increasing, zero, decreasing}

L has domain: {full, empty, partial}

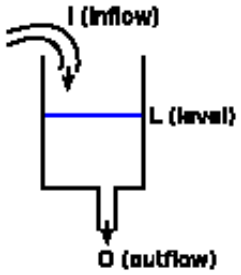

Figure 1: Water Level Model

I has domain: {zero, high, low}
O has domain: {zero, high, low}

Seek to come up with discrete values to model a continuous and dynamic system. Then come up with a discrete constraint model for the system. Can be extended to multi-tank systems.

The goal was to come up with simple qualitative models of the world.

6 Lecture 31 January 2007

Simplification of Physics led to Qualitative Physics. A physics model of the tank would include A the cross-section of the tank and O the cross-section of the outlet. The change of height over time: $\frac{\delta h}{\delta t} = \frac{Q_{in}}{A} - \frac{O}{A}\sqrt{2gh}$. This is quite difficult.

$\dot{h} = M^+(Netflow)$ is the simplified version from Qualitative Physics.

Qualitative concepts:

- $N = Netflow = Q_{in} - Q_{out}$.
- M^+: Monotonically increasing. $y = M^+(x)$: y is proportional to x.
- M^-: Monotonically decreasing. $y = M^-(x)$: y is inversely proportional to x.

Redefine Parameter Space:

For example the flow Q_{in} and Q_{out} can be mapped from a continuous domain to $\{+, 0, -\}$ discrete values. So $D[Q_{in}] = \{+, 0\}$, $D[Q_{out}] = \{+, 0\}$ and $D[N] = \{+, 0, -\}$.

Performing Inference: Using Qualitative Algebra and Constraints over that algebra.

$N = Q_{in} - Q_{out}$:

Q_{in}	Q_{out}	
	+	-
+	?	+
-	-	0

Two problems with Quantitative Modelling:

1. Indeterminate Values (e.g. "?" in above table).
2. Complexity (turns out to be NP-hard).

6.1 Example Model

Take a two tank system with the following constraints:
$Q_1 = Q_2 + Q_3$ and $Q_3 > Q_4$.

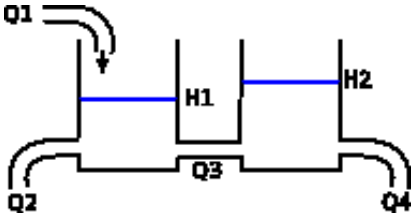

Figure 2: Two Tank System

Analysis of Decomposition

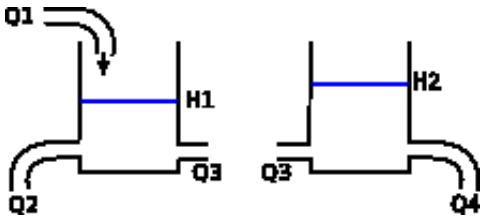

Figure 3: Decomposed System

$Q_1 = \{+, 0\}$; $Q_2 = \{+, 0\}$; $Q_3 = \{+, 0, -\}$; $Q_4 = \{+, 0\}$.
$\dot{h} = M^+(Q_3 - Q_4)$.
However, constraint $Q_3 > Q_4$ may not be physically realisable.

7 Lecture 1 February 2007

Qualitative Analysis is used in Computer-Generated Graphics Animation (in films and games). Also used in schooling in the US - driven by SAT testing.

7.1 History

1970s: Heuristic Approaches - Commonsense Reasoning; View everything qualitatively and use heuristic algorithms. E.g. Medical diagnosis expert systems.

1980s: Start of Model-Based diagnosis - Formal Modelling; Logic-Based Models and Probabilistic Models.

1990s: State of the Art models - Tunable algorithms and models; Embedding models in real systems (e.g. NASAs Deep Space One had an AI running the spaceship).

2000s: Application Areas: Drug Discovery - Models of Metabolic system.

7.2 Diagnostic Problem Solving

Diagnosis model is always a super-set of simulation model.

Two components: SD (System Description); OBS (Observation).

SD includes embedded sensors which monitor the mode of the system (i.e. fault status of each component).

Example of a logic buffer.

Simulation Model:

In	Out
1	1
0	0

Diagnosis Model of same system:

Mode	In	Out
OK	1	1
OK	0	0
Invert	1	0
Invert	0	1
Stuck 0	1	0
Stuck 0	0	0
...		

How do you determine model fidelity?

Conservatism ensure that rule-based systems predominate. The system monitoring the main engines on the Space Shuttle (a highly complex and critical system) has nine (count them 9) rules!

8 Lecture 7 February 2007

Projects can be Domain Analysis or Theory. Model a real system (e.g. home & business security) and analyse it. Will recommend tools and scope. One-to-one meetings next Monday.

I have suggested Spam Filtering analysis using Bayesian Networks (GeNIe modelling tool).

9 Lecture 8 February 2007

VLSI Very Large-Scale Integration, a process for the creation of electronic integrated circuits.

ATMS Assumption-based Truth Maintenance System, a system for maintaining consistency in a model.

SD System Description (the model).

AB() Abnormal.

MBD Model-Based Diagnosis.

- Component Oriented
  - Structural Models
  - Functional Models
  - Behavioural Models
    - Correct Behaviour
    - Fault Models
- Process Oriented
  - Causal Models
  - Process Models

Behavioural Models can be: Static or Dynamic; Quantitative or Qualitative; etc.

10 Lecture 14 February 2007

Project and Presentation
Formulate approach to finding solutions to open-ended questions.

1. Literature survey (google scholar)
2. Scoping (focus on important/relevant portion of problem/complex system.

Presentation: Literature Survey and Scope (10-15 min; next Thur and Wed).

11 Lecture 15 February 2007

Presentation: Here is the area of the project and what I will contribute.

11.1 Model-Based Diagnosis

On detecting discrepancies between expected and observed results, infer where the error may have arisen (and which components cannot be contributing to the fault).

Masking Faults fault which makes things appear normal.

Hitting Set minimal subsets which could cause observed error.

Consistency-Based Diagnosis (Reiter 1987):
Assume components normal.
SD: System Description (the model)
COMPS: The set of components
OBS: The observations
Observed discrepancy $\Rightarrow$ Search for minimal set of faulty components.
$SD \cup OBS \cup \{AB(c)|c \in \Delta\} \cup \{not\}\, AB(c)|c \in COMPS - \Delta$ is consistent.
Testability is dual to this framework - find the minimal number of tests such that you can isolate faults (then run these test continuously). Finding the set of tests is an NP-hard problem.

11.1.1 Probability-Based

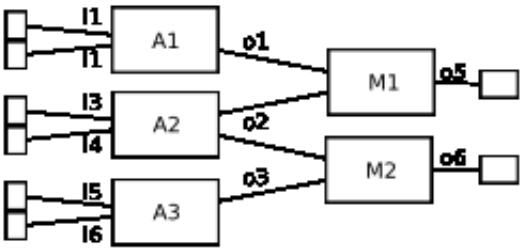

Figure 4: Logical Diagram

Graph Theoretic Framework

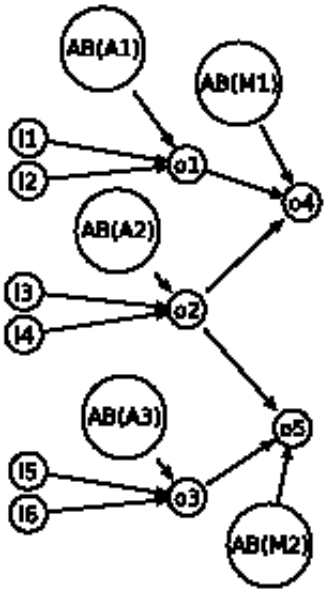

Figure 5: Causal Diagram

Shows dependence and independence.

12 Lecture 28 February 2007

12.1 Bond Graph

Bond Graph approach came out of engineering.

A bond graph is a graphical description of a physical dynamic system. It is an energy-based technique. A bond graph shows the energy flows around a system. Bond graphs have a number of advantages over conventional block diagram or code-based modelling techniques:

- They provide a visual representation of the system.
- Since they work on the principle of conservation of energy, it is difficult to accidentally introduce extra energy into a system.
- The bonds are symbols which contain meaning.
- They separate the structure from causality.
- Since each bond represents a bi-directional flow, systems which produce a 'back force' (e.g. a motor back emf) on the input are easily modelled without introducing extra feedback loops.
- Hierarchy can be used to manage large system models.

Bond graphs are based on the principle of continuity of power. If the dynamics of the physical system to be modeled operate on widely varying time scales, fast continuous-time behaviors can be modeled as instantaneous phenomena by using a hybrid bond graph.

12.2 Modelling

Generic Approach:

1. Start with a Schematic of the Complex system
2. Decompose into hierarchical Block Diagram (Causal Diagram). These diagrams are function neutral - could use Formal Logic or Probability
3. Convert the Block Diagram into a Bond Graph.
4. Transform the Bond Graph (via formal transformations) into:
   (a) Ordinary Differential Equation Models.
   (b) Partial Differential Equation Models.
   (c) Qualitative Physics Models.
   (d) etc.

12.2.1 Bond Graph Elements

The constitutive equations of the bond graph elements are introduced via examples from the electrical and mechanical domains. The nature of the constitutive equations lay demands on the causality of the connected bonds. Bond graph elements are drawn as letter combinations (mnemonic codes) indicating the type of element. The bond graph elements are the following:

C storage element for a q-type variable, e.g. capacitor (stores charge), spring (stores displacement).

I storage element for a p-type variable, e.g. inductor (stores flux linkage), mass (stores momentum).

R resistor dissipating free energy, e.g. electric resistor, mechanical friction.

Se Sf energy and flow sources, e.g. electric mains (voltage source), gravity (force source), pump (flow source).

TF transformer, e.g. an electric transformer, toothed wheels, lever.

GY gyrator, e.g. electromotor, centrifugal pump.

0 1 0- and 1-junctions, for ideal connecting two or more sub-models.

13 Lecture 14 March 2007

Mapping representations to algorithms.

There is a trade-off between computational complexity and representational detail. The complexity tends to increase exponentially as the amount of detail in a model is increased.

Different types of representation have inherently different amounts of complexity. Rule-Based systems are efficient but limited in their expressibility.

Representing a system as a graphical model tells us a lot about its complexity.

Rule-based systems which can be represented as Single Connected Directed Graphs (i.e. where there is only one directed path between any pair of nodes) are of P-complexity. (i.e. can be solved in Polynomial time).

If a Directed Acyclic Graph (DAG) is needed then it is of NP-complexity (Non-deterministic Polynomial).

All of the different types of models covered can be represented by graphs (e.g. Constraint Satisfaction Problems by Constraint Graphs).

There is a point when increasing detail results in a dramatic increase in computational complexity. This is known as a *phase transition*.

13.1 Graphs

Directed graph (or digraph) $G = (V, E)$ consists of a finite set V, called vertexes or nodes, and E, a finite set of ordered pairs, called edges of G. E is a binary relation on V. Cycles, including self-loops are allowed. Multiple edges are not allowed though; (v, w) and (w, v) are distinct edges.

Undirected graph (or simply a graph) $G = (V, E)$ consists of a finite set V of vertexes, and a finite set E of unordered pairs of distinct vertexes, called edges of G. Cycles are allowed, but not self-loops. Multiple edges are not allowed.

- Vertex v is *adjacent* to vertex u if there is an edge (u, v).
- Given an edge $e = (u, v)$ in an undirected graph, u and v are the *endpoints* of e, and e is *incident* on u and on v.
- In a digraph with edge $e = (u, v)$, u and v are the *origin* and *destination*. We say that e leaves u and enters v.

- A digraph or graph is *weighted* if its edges are labeled with numeric values.
- In a digraph,
  - The *Out-degree* of v is the number of edges coming from v.
  - The *In-degree* of v is the number of edges coming into v.
- In a graph, the *degree* of v is the number of edges incident to v. (The in-degree equals the out-degree).
- *Path*: a sequence of vertexes $\langle v_0, \ldots, v_k \rangle$ such that (v_{i-1}, v_i) is an edge for $i = 1$ to k, in a digraph. The length of the path is the number of edges, k.
- w is *reachable* from u if there is a path from u to w. A path is *simple* if all vertexes are distinct.
- *Cycle*: a path in a digraph containing at least one edge and for which $v_0 = v_k$. A cycle is *simple* if, in addition, all vertexes are distinct.
- For graphs, the definitions are the same, but a *simple cycle* must visit ≥ 3 distinct vertexes.
- An *Eulerian cycle* is a cycle, not necessarily simple, that visits every edge of a graph exactly once.
- A *Hamiltonian cycle* (or *path*) is a cycle (path in a directed graph) that visits every vertex exactly once.

A family in a graph consists of a node and all of its parents.

For Bayesian networks the fastest computers (2007) can deal with graphs with family sizes of up to 25. Up to size 5 graphs are easy, by size 20 the systems struggle.

Complexity is $O(2^k)$ where k is the size of the clique. A clique is a set of nodes who are all connected to one another. Families can be transformed into Cliques.

For Bayesian Networks: Using such simple mechanisms we can say how computationally complex the model is.

It is much more difficult to predict the computational complexity for other representations.

In a logical system: n variables indicates the complexity of inference is $O(n)$. Then there are the additional issues of the structure of the logical propositions. 2-SAT (satisfiability problem with two variables per clause is $O(P)$; while 3-SAT is $O(NP)$.

Graph Algorithms generally fall into two classes:

1. Polynomial Time Algorithms (e.g. Depth First Search)
2. General Algorithms (e.g. Tree Decomposition)

The connectivity of the graph plays a role in determining the complexity of algorithms which solve the problem.

13.2 Complexity

The complexity class P is the set of decision problems that can be solved by a deterministic machine in polynomial time. This class corresponds to an intuitive idea of the problems which can be effectively solved in the worst cases.

The complexity class NP is the set of decision problems that can be solved by a non-deterministic machine in polynomial time. This class contains many problems that people would like to be able to solve effectively, including the Boolean satisfiability problem, the Hamiltonian path problem and the vertex cover problem. All the problems in this class have the property that their solutions can be checked efficiently.

Problems are grouped into "classes" depending on their runtime.

P class Problems that can be solved in polynomial time. They are $O(n^k)$ problems, where n is the input size, and k is some constant not dependent upon n. Examples: Depth-First Search; Breadth-First Search; All BST operations; All Fibonacci, Binomial, B Tree, B+ Tree operations; All linked list operations etc..

All are at most polynomial time operations.

The class represents those problems that are known to have an algorithm that is bounded within polynomial time. Polynomial time means that the computational complexity is bounded by $O(f(n))$ where $f(n) = n^k$ for some constant k. (This includes problems bound by $O(log\, n)$, $O(n)$, $O(n^2)$, $O(n^8)$, even $O(n^{1000})$! This does not include problems bound by $O(2^n)$ or $O(n!)$.)

Why should we be interested in this class if it can have such a great range? Because it generally tells us that if a problem is not in P, then it is too hard to solve (as its solution will be at least $O(2n)$)

Problems with large computational complexities are often referred to as intractable, meaning that it is not realistic to try to solve them. Tractable problems are those that can be done in a reasonable amount of time on modern computers - some problems in P may not be considered tractable, but all problems not in P are considered intractable

NP class If you have a solution to a problem, is there a polynomial-time algorithm that can verify that the solution is indeed a solution to the problem? If so, then the problem is in NP.

14 Lecture 15 March 2007

The set of NP-complete problems are mutually reducible (all can be transformed into all).

No Free Lunch: If a hard problem can be transformed into one which is easy to solve then the transformation will be hard. There seems to be a law of conservation of complexity in operation.

Knowing how complex a specific question will be to answer (including to discover that there is no answer) is important when deciding what algorithms to use.

For very complex cases an approximation algorithm is the more likely approach. If an approximate answer is not acceptable then the problem specification will need to be relaxed in some way.

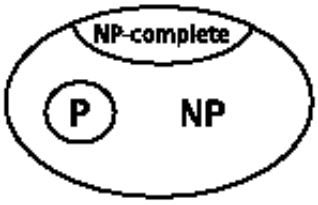

Figure 6: Complexity Classes

15 Lecture 21 March 2007

On this module we are interested in declarative representations. where there is a precise mapping between reality and the model. (This module is not about Machine Learning).

When building a system to play chess, in general, the problem is broken down into three parts, with different representations for each:

1. Opening (Database of Openings and Counters)
2. Mid-game (Game Tree and search - AI Search and Machine Learning)
3. End-game (Simulation)

The first and last parts are quite easy. The mid-game is more difficult as human Grandmaster methods have not been encoded.

15.1 Bayesian Inference

Bayesian Networks

Structured, graphical representation of probabilistic relationships between several random variables.

Explicit representation of conditional independencies.

Missing arcs encode conditional independence.

Efficient representation of joint probability dependencies.

Allows arbitrary queries to be answered.

Knowledge Acquisition

- Variables:
  - collectively exhaustive, mutually exclusive values
  - clarity test: value should be knowable in principle
- Structure
  - if data available, can be learned

  - constructed by hand (using "expert" knowledge)
  - variable ordering matters: causal knowledge usually simplifies

- Probabilities
  - can be learned from data
  - second decimal usually does not matter; relative probabilities
  - sensitivity analysis

Microsoft make extensive use of Bayesian Networks in their help tools.

16 Lecture 26 March 2007

16.1 Tree decomposition

Convert an arbitrary graph into a tree (of cliques).

1. Moralisation, marry (link) unconnected parents.
2. Triangulation, in any cycle of length greater than three, every triple must have a chord. Create a set of triangles.
3. Identify cliques.
4. Generate Clique tree, maintain the running intersection property.

running intersection property For each node v of G, the cliques and separators containing v form a connected subtree of T.

Inference is closely ties to representation. Some classes of representation make inference easy. If the topology of the graph representing the problem is a tree then inference is $O(n)$; For multiply connected graphs it is NP-Hard.